LAUREA
AMMATTIKORKEAKOULU
Yhdessä enemmän

Tietoturvatarkastus web-sovellukselle

Matias Koskinen

2018 Laurea

Laurea-ammattikorkeakoulu

Tietoturvatarkastus web-sovellukselle

Matias Koskinen
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Kesäkuu, 2018

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittelyn koulutusohjelma
Tietojenkäsittelyn koulutusohjelma (AMK)

Matias Koskinen

Tietoturvatarkastus web-sovellukselle

Vuosi 2018 Sivumäärä 31

Tämän opinnäytetyön tarkoitus oli käydä läpi tyypillisen web-sovelluksen tietoturvan tarkastaminen. Työn tavoitteena oli tutkia tyypillisimmät web-sovellusten haavoittuvuudet ja esitellä niihin korjausehdotukset alan parhaiden ratkaisuiden mukaisesti. Lisäksi tavoite oli antaa esimerkit, miten tarkastaa löydetyt haavoittuvuudet sekä havainnollistaa, miten hyökkääjät voivat niitä käytännössä hyväksikäyttää. Lopputuotoksen suhteen tavoitteena oli luoda yrityksille, harrastajille sekä opiskelijoille hyödyllistä materiaalia ja tarjota materiaalia kymmenen suurimman haavoittuvuuden ymmärtämiseen ja torjuntaan.

Työn tietoperustassa esitellään tärkeimpiä työkaluja, joilla web-sovellusten tietoturvatestausta voidaan suorittaa. Tietoperustan ovat muodostaneet alan kirjallisuus ja internet-julkaisut sekä kokemuksen myötä kertynyt oma ammattitaito. Keskeisenä lähteenä on ollut OWASP:in Top 10 – 2013 The Ten Most Critical Web Application Security Risks –julkaisu.

Työn tuloksena syntyi ohjeistus web-sovellusten tarkastamisen ja auditoinnin suorittamiseen. Ohjemateriaali soveltuu myös itseopiskeluun ja perehdytykseen. Asiakasyritykselle tehtävää web-sovelluksen tarkastusta ei ole liitetty tähän opinnäytetyöhön tulosten osalta.

Johtopäätöksinä kootaan yhteen yritysten yleisimmät puutteet web-sovellusten tietoturvallisuudessa ja esitetään näihin korjausehdotukset sekä korjausehdotusten tueksi annetaan esimerkkejä puutteiden hyväksikäytöstä.

Avainsanat: Tietoturva, web-sovellus, ohjeistus, teknologia, IT

Laurea University of Applied Sciences **Abstract**
Degree Programme in Business Information Technology
Bachelor's Degree of Business Information Technology

Matias Koskinen

A Security Audit for a Web Application

Year 2018 Pages 31

The objective of this thesis was to conduct the security assessment of a typical web application. The goal was to inspect the most typical vulnerabilities in web applications and present the plans on how to fix those vulnerabilities according to the best practices in the field. In addition, an aim of this study was to give examples on how to assess the vulnerabilities that were found and to demonstrate how an attacker could exploit these vulnerabilities. The goal of the final product was to offer useful material for companies, hobbyists and students to understand and deflect the ten greatest vulnerabilities.

In the theoretical part of this thesis the most important tools for web application assessment are presented. The knowledge base for this thesis consists of literature on cybersecurity, internet publications and the expertise of the writer. A crucial source of information is OWASP's "Top 10 – 2013 – The Ten Most Critical Web Application Security Risks" –publication.

As a result a study material for cybersecurity oriented and especially web application security oriented people is created. The final product is a guide on how to assess the security of web applications.

As a conclusion the most common shortcomings in web application security, the proposed correction instructions for these shortcomings and examples how to exploit them are presented.

Keywords: Cybersecurity, web application, how-to, technology, IT

Sisällys

1 Johdanto

Tämän opinnäytetyön aiheena on ohjeistus web-sovelluksen perustason tietoturvatarkastukseen. Ohjeistus koostuu yleisimpien haavoittuvuuksien olemassaolon syiden sekä niiden hyväksikäytöstä johtuvien seurausten selvittämisestä. Tämän lisäksi haavoittuvuuksille esitetään yleisellä tasolla korjausehdotuksia, jotka implementoimalla voidaan poistaa haavoittuvuuksien aiheuttamat uhat. Yleisen selvityksen ja korjausehdotusten lisäksi esitellään vielä jokaiselle haavoittuvuudelle toisto-ohjeet, jotta voidaan todeta haavoittuvuuden olevan korjattu.

Haavoittuvuuksien testauksen ja korjausohjeistuksen lisäksi tutkitaan web-sovellusten testaukseen luotuja ja testausprosessissa hyödyllisiä testaustyökaluja. Käsiteltävät työkalut ovat keskeisessä roolissa peruslaatuisessa web-sovelluksen testauksessa ja osaa niistä hyödynnetään opinnäytetyötä tehdessä haavoittuvuuskartoituksessa.

Web-sovellusten yleisimmät ongelmat on määritelty käyttäen tietoturva-alan osaajien laatimaa OWASP Top 10 -haavoittuvuuslistausta. Opinnäytetyöhön sisältyy teknisten ratkaisujen tarkastamisen lisäksi pohdintaa ja arviointia ongelmien vakavuudesta. Web-sovellustarkastuksen esimerkkikohteena geneeristä web-sovellusta, joka voisi olla käytössä pienellä yrityksellä. Opinnäytetyössä käytetään lähteinä web-sovellusteknologioihin liittyviä dokumentteja kyseisten teknologioiden parhaista tietoturvaratkaisuista (cybersecurity best practices). Tietotekniset termit selitetään auki alan kirjallisuutta ja sähköisiä lähteitä lainaten.

Opinnäytetyön alkuun on liitetty sanalista kaikista alan termeistä, jotka esiintyvät muissa kappaleissa. Varsinainen aiheen tutkiminen alkaa selvityksellä siitä, mitä tietoturva pääpiirteisesti pitää sisällään ja mistä siinä on kyse. Sen jälkeen käyn läpi miksi tietoturva on tärkeää ja minkälaiset syyt ovat nostaneet sen kasvavaksi alaksi. Seuraavaksi kerron, mitä websovellukset ovat ja miten ne rakentuvat sekä selitän karkealla tasolla miten ne toimivat.

Tietoturvauhkien listauksen apuna tulen käyttämään OWASPin web-sovellusten haavoittuvuuslistausta. Tulen esittelemään opinnäytetyössä myös mielestäni parhaiten web-sovellusten tietoturvatestaukseen soveltuvia työkaluja, joilla voidaan tehokkaasti kartoittaa ja löytää haavoittuvuuksia web-sovelluksista.

2 Tietoturva

Tietoturva on laaja-alainen kokonaisuus, joka jakaantuu useaan osaan. Keskeinen osa tietoturvaa on niin kutsuttu CIA triadi. Se perustuu ideaan, jonka mukaan tiedon luottamuksellisuus (engl. confidentiality), eheys (engl. integrity) sekä saatavuus (engl. availability) tulee olla turvattu. Luottamuksellisuus tarkoittaa, että arkaluontoista tietoa pystyvät käsittelemään vain ne tahot, joilla siihen on oikeus. Luottamuksellisuuden epäonnistuessa kyse on lähes poikkeuksetta tietovuodosta. Tiedon eheyden säilyttäminen tarkoittaa sitä, että varastoitu tieto säilyy muuttumattomana, ja vain hyväksytyt tahot pystyvät muuttamaan tietoa. Tiedon muuttaminen luvatta aiheuttaa epävarmuutta tiedon oikeudesta sekä nakertaa luottamusta tietoon. Tiedon saatavuus puolestaan tarkoittaa, että tiedon tulee olla saatavilla käyttäjilleen. Yleisesti tiedon saatavuutta voidaan loukata esimerkiksi palvelunestohyökkäyksellä, joka estää käyttäjien pääsyn tietoon. (Studdard & Pinto, 2011.)

3 Tietoturvan tärkeys

Tietoturva ei turhaan ole tällä hetkellä nouseva ala. Internetiin kytkettävien laitteiden määrä lisääntyy koko ajan. Kun yhä laajempi ja laajempi kirjo laitteita on alkanut saada perusominaisuuden ottaa yhteyden internetiin, on syntynyt termi internet-of-things eli IoT. Suomeksi tämä termi tarkoittaa esineiden ja asioiden internetiä.

Valmistajat ovat perinteisesti tottuneet valmistamaan tuotteita kuluttajille mahdollisimman edullisesti ja mahdollisimman monipuolisilla ominaisuuksilla varustettuina. Edullinen hinta on vaatimus, joka saa aikaan kilpailua ja halpaa hintaa jahdatessa leikataan usein asioista, jotka eivät ole kuluttajalle ominaisuutena ensiarvoisen tärkeitä, kuten tietoturvallisuus.

Tämä johtaa siihen, että ihmisten yksityisyys ja muutoin reitittimien palomuurien ja tietoturvaohjelmistojen suojaamat suhteellisen turvalliset sisäverkot alkavat vaarantua heikoimman lenkin periaatteella. Kun internetiin kytketty vedenkeitin paljastaa salasanoja selväkielisenä, on helppoa päästä turvatoimien ohitse.

IoT ei ole ainoa asia miksi tietoturva on painava aihe. Vielä muutama vuosikymmen sitten kaikki henkilön elämään vaikuttava hallinnointi kuten pankki- ja verkkokauppa-asiointi tehtiin kasvotusten. Nykyään verkkopankit ja verkkokaupat ovat kätevin tapa hoitaa asiat. Kun henkilön omia tietoja siirretään internetin välityksellä, herää kysymys kuka näihin tietoihin pääsee käsiksi ja millaiset vaikutukset tietojen paljastumisella on henkilön elämään. Tästä on hyvänä esimerkkinä vuonna 2015 sattunut tietomurto, joka tapahtui avioliiton ulkopuolisia suhteita mahdollistavalla sivustolla nimeltä Ashley Madison. Palvelun käyttäjien tietoja vuoti julkisuuteen ja avioliiton ulkopuolisia suhteita harrastavien henkilöiden tiedot olivat kaikkien saatavilla. (A timeline of the Ashley Madison hack 2017.)

Tietovuodot vaurioittavat yritysten mainetta ja voi aiheuttaa jopa yhtiön osakkeen arvon vaihtelua. Yksityishenkilöille salaisen tiedon paljastuminen voi tarkoittaa työpaikan, ystävien tai poliittisesti kireissä maissa jopa hengen menetystä.

4 Web-sovellukset

Webillä tarkoitetaan verkkoa eli internetiä. Internet on laaja tietokoneiden ja muiden laitteiden verkko. Web-sovellus on tapa tuoda palvelimelta toimintoja saataville internetin käyttäjille. ”Web-sovellus on se kokonaisuus, johon verkkoselain ottaa yhteyden kommunikoidakseen verkkopalvelimen kanssa.” (Studdard & Pinto,1.) Jokainen web-sovellus, joka on saatavilla internetissä, toimii siis fyysisellä palvelimella jossakin päin maailmaa ja kun selaimen ikkunaan ilmestyy sisältöä, se sisältö on haettu tuolta palvelimelta.

Web-sovellusten tarina alkaa hyvin yksinkertaisista verkkosivuista, joihin verkkosivun omistaja on voinut kirjoittaa vaikkapa päiväkirjaansa sekä arkistoida vanhat päiväkirjamerkinnät luettavaksi myöhemmin. Perinteinen esimerkki on henkilön itselleen tai vaikkapa koiralleen luomat verkkosivut, joissa on tekstiä ja kuvia. Verkkosivuilla alkuun, kuten nykyäänkin on voinut olla perinteisesti sinisiä alleviivattuja hyperlinkkejä, joilla on pystynyt navigoimaan sivuston sisällä tai sieltä jollekin toiselle sivulle. Tämän kaltainen luettavan materiaalin ja kuvien tarjoaminen ja niiden välillä navigointi on ollut aikaisimpia toiminnallisuuksia, joista nykyiset web-sovellukset ovat kehittyneet.

Nykyaikaisissa web-sovelluksissa on useita tasoja, sekä paljon enemmän toiminnallisuuksia. Nykyisin verkkosivuille on mahdollista sisällyttää esimerkiksi JavaScript –koodia, joka avaa täysin uudenlaisen maailman kuin mistä ensimmäiset verkkosivut aloittivat. Nykyisin web-sovellukset hyödyntävät useassa sijainnissa olevia skriptejä eri ominaisuuksien tuottamiseen näkyville sovelluksen käyttäjälle ja yhden sovelluksen toiminta riippuu usean eri komponentin yhteistoiminnasta. Sovelluksen taustalla toimii palvelin, joka säilöö sovelluksen käsittelemiä tietoja ja useissa tapauksissa myös käyttäjän syöttämiä tietoja, joita käyttäjä voi myöhemmin lukea palvelimelta käyttäen web-sovellusta. (Studdard & Pinto 2011.)

Web-sovellusten toiminnan keskeisessä osassa ovat HTTP-viestit. HTTP-viestien avulla selain (client) viestii palvelimen (server) kanssa. HTTP-viestejä on kahta tyyppiä, kutsut ja vastaukset. Kutsut (HTTP request) ovat HTTP-viestejä, joita selain lähettää palvelimelle pyytäen tai ’kutsuen’ jotakin resurssia palvelimelta. Vastaus (HTTP response) on HTTP-viesti, jonka palvelin lähettää selaimelle vastaten selaimen lähettämään kutsuun. (Mozilla 2015.)

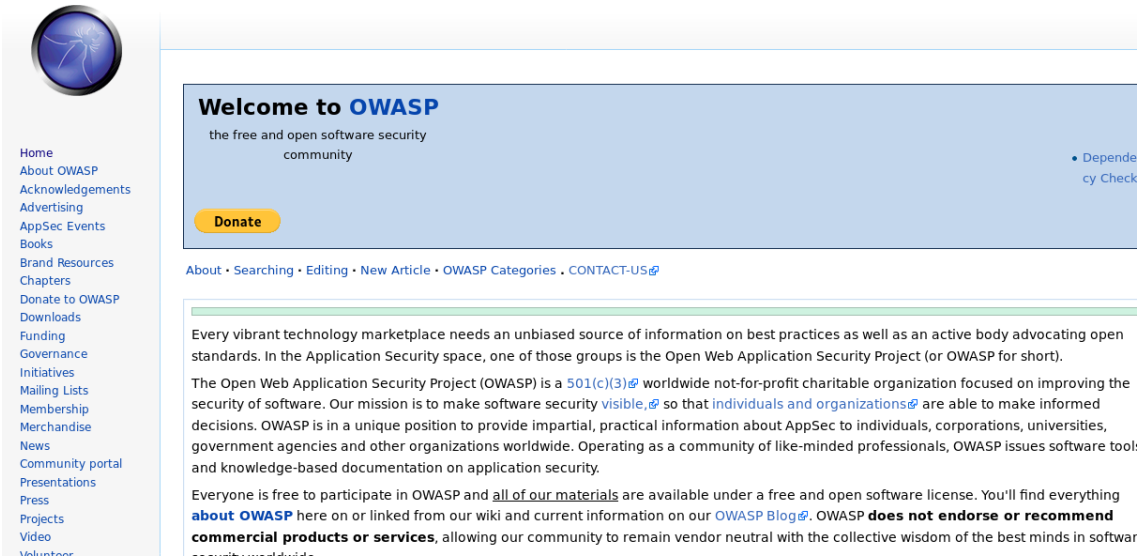
Kutsujen tarkemmat toiminnot määritetään HTTP-metodeilla (HTTP method). Metodi määrittää minkälainen toimenpide kutsutulle resurssille suoritetaan. Yleisimmät verkkoselaamiseen

liittyvät metodit ovat GET ja POST. GET -metodi sananmukaisesti noutaa resurssit palvelimelta ja sitä käytetään esimerkiksi verkkosivun lataamiseen. Mikäli haluat ladata iltapäivälehden etusivun palvelimelta, lähetät palvelimelle kutsun GET ja näin ohjeistat palvelimen lähettämään resurssin selaimellesi.

Resursseja noutavaan GET -metodiin verrattuna POST -metodi on päinvastainen toiminto. Lähettämällä POST -kutsun, palvelin ottaa vastaan kutsun loppuosaan lisätyn datakokonaisuuden (data entity). Datakokonaisuus voi olla esimerkiksi merkkijono, joka sisältää käyttäjän käyttäjätunnuksen sekä salasanan.

5 OWASP Top 10

OWASP eli avoin web-sovellusten turvallisuusprojekti (Open Web Application Security Project) on avoin yhteisö, joka on omistautunut mahdollistamaan organisaatioille luotettavien sovellusten kehittämisen, oston sekä ylläpidon. (Owasp Top 10 – 2013 2013.)



Kuva 1 OWASPin kotisivu

OWASP on määritellyt alan suurimmat kymmenen haavoittuvuutta, jotka on luokiteltu järjestykseen vakavuuden ja yleisyyden mukaan. OWASPin Top 10 -lista heijastaa ajankohtaisesti yleisimmät ja vakavimmat uhat web-sovelluksille ja tämän vuoksi kyseinen lista on arvostettu alan ammattilaisten keskuudessa. Tässä opinnäytetyössä käytetään viitemateriaalina OWASP Top 10 vuoden 2013 versiota, sillä tätä opinnäytetyötä laatiessa vuoden 2017 versio on vielä keskeneräinen.

OWASP on luonut web-sovellusten lisäksi myös esimerkiksi mobiiliapplikaatioille oman Top 10 –listauksensa ja sekaannusten välttämiseksi kussakin listauksessa käytetään etuliitteenä kirjainta, joka kuvaa mitä alustaa tarkastellen haavoittuvuudet ovat listattu. Web-sovelluksiin liittyvässä listauksessa tunnuskirjaimena toimii A, joka johtaa merkityksensä englannin kielen sanasta ”application”, kun taas esimerkiksi mobiilisovellusten listauksessa käytetään kirjainta M.

Tässä osiossa käsitellään OWASPin Top 10 luettelon sisältämät haavoittuvuudet. Niiden perusolemus selostetaan auki kuvaavasti ja selvitetään mistä niissä on kyse. Jokaisessa haavoittuvuudessa selvitetään mistä ne johtuvat, mitä vaaroja ne aiheuttavat ja miksi ne tekevät niin. Perusselvityksen jälkeen esitellään tapoja, joilla näitä haavoittuvuuksia voi itse kokeilla esimerkiksi harrastajan omassa kotiympäristössä sijaitsevassa testisovelluksessa. Lopuksi esitellään korjausehdotukset näille haavoittuvuuksille sellaisella tavalla, jota seuraamalla on helppo todeta, ettei haavoittuvuuden hyväksikäyttö ole enää mahdollista.

Alla listattuna OWASP Top 10 –listauksen kymmenen web-sovellusten yleisintä ja vaarallisinta haavoittuvuutta.

5.1 Injektiot

Injektioiden toimintaperiaate on, että web-sovelluksen sisältämiin syötekenttiin syötetään sisältöä, kuten koodia, jonka kohdesovellus käsittelee ja suorittaa antaen syötteen tekijälle halutun vastauksen tai toiminnallisuuden suorituksen. Tästä voidaan käyttää esimerkkinä perustavan laatuista SQL-injektiota. SQL (Structured Query Language) on koodikieli, jolla voidaan käyttää tietokantoja. (Introduction to SQL 2018.)

Ymmärtääkseen SQL-injektiota paremmin, täytyy ymmärtää SQL-kieltä. SQL-kieli sisältää kahdenlaisia lausekkeita: DDL-lausekkeita (Data Definition Language) eli datan määrittämiseen eli esimerkiksi luomiseen käytettyjä lausekkeita ja DML-lausekkeita (Data Manipulation Language) eli datan manipuloimislausekkeita. Datan manipulointi voi olla esimerkiksi sen hakemista tietokannasta. (SQL query structure 2013.)

Tässä tapauksessa keskitymme DML-lausekkeisiin, sillä dataa manipuloimalla hyökkääjä voi tietoja haltuunsa. SQL-kyselyt, joita SQL-kielellä pääosin tehdään, muodostuminen on kuvattu seuraavassa kuvassa. Kaksi ylintä esimerkkiä kuvassa esittävät teoriapohjaa SQL:n taustasta ja alimmainen esimerkki on esimerkki oikeasta SQL-kyselystä. (Kuva 2)

Kuvan SQL-lauseke muodostuu sanasta SELECT, joka tarkoittaa, että kyselyn seurauksena valitaan resurssi. Tässä esimerkissä resurssiksi on määritelty tähtimerkki (*), joka tarkoittaa että SQL-kysely haluaa palautettavaksi kaikki tiedot. Seuraavaksi määritellään FROM, joka määrittää taulun (table), josta kaikki tiedot valitaan. Esimerkissä tauluksi on määritetty käyttäjät. Tähän asti käsitelty lauseke ”SELECT * FROM käyttäjät” toimisi jo itsessäänkin, mutta on syytä

määritellä vielä lisäksi ehto, jotta saadaan vain tietyn käyttäjän tiedot. SQL-lausekkeeseen lisätään sana WHERE, jonka jälkeen määritellään ehto. Tarkoituksena on noutaa esimerkkikäyttäjä Jukan tiedot, joten se määritellään taulusta käyttäjät haettavaksi kaikki tiedot kohdasta, jossa parametri nimi on Jukka.

```
SELECT jotain FROM jostakin taulusta
     WHERE jotkin ehdot täyttyvät

UPDATE jokin taulu SET jokin
     WHERE jotkin ehdot täyttyvät

SELECT * FROM käyttäjät
     WHERE nimi='Jukka'
```

Kuva 2 SQL-kielen rakenne

SQL-kyselyn kohtaan, jossa määritellään ehtolause, voidaan lisätä useita ehtoja lisäämällä perään erottamalla ehdot toisistaan sanalla AND tai OR. SQL-injektiossa hyödynnetään tätä. Mikäli web-sovellusta ei ole suojattu SQL-injektiota vastaan, hyökkääjän on mahdollista suorittaa SQL-kyselyitä esimerkiksi sisäänkirjautumiseen tarkoitetussa kirjoituskentässä.

Perinteinen SQL-injektio syötetään sisäänkirjautumissivulla käyttäjäkenttään haluttu käyttäjätunnus ja salasanakenttään SQL-ehtolauseke ”diibadaaba OR ’1’=’1”. Sovellus, jossa ei ole minkäänlaista suojaa SQL-injektioita vastaan, tulkitsee ehtolauseesta salasanan olevan oikea, sillä toinen SQL-kyselyssä esitetyistä ehdoista käy toteen. Salasana on joko ”diibadaaba” tai vaihtoehtoisesti arvo 1 on yhtä suuri kuin 1. Sisäänkirjautuminen tapahtuu, sillä salasanan tarkistava osa web-sovellusta tulkitsee salasanan vääräksi mutta arvon 1 yhtäsuureksi arvon 1 kanssa, sillä riittää että vain toinen ehto on tosi.
Tällä tekniikalla hyökkääjän on mahdollista ohittaa kirjautuminen ja kirjautua sisään esimerkiksi järjestelmänvalvojana vaihtamalla käyttäjätunnuksen kohdalle järjestelmänvalvojan käyttäjätunnus.

Injektiohaavoittuvuuden ovat listan kärjessä hyvästä syystä, sillä tämän tasoinen kirjautumisen ohittaminen aiheuttaa erittäin suuren turvallisuusuhan, etenkin järjestelmänvalvojan oikeuksilla. Kirjautumisen ohittamisen lisäksi injektiohaavoittuvuuksilla voidaan saada arkaluontoista tietoa vaikkapa sovelluksen taustalla toimivasta tietokannasta. Hyökkääjä voi syöttää haavoittuvaan sovellukseen koodia, joka noutaa tietokannasta käyttäjien kaikki tiedot ja listaa ne hyökkääjän nähtäville. Arkaluontoisia tietoja voivat olla esimerkiksi asiakastiedot sekä pankkikorttien tiedot.

5.2 Rikkinäinen autentikaatio ja istunnonhallinta

Web-sovellusten rikkinäinen autentikaatio ja istunnonhallinta mahdollistaa hyökkäyksiä sovelluksen tietoturvaa kohtaan. Hyökkääjä voi saada haltuunsa salasanoja, avaimia tai istuntotunnisteita tai muutoin käyttää hyväkseen heikkoja autentikaation toteutustapoja. Näillä tavoin hyökkääjä voi omaksua sovelluksen käyttäjien tai jopa järjestelmänvalvojien identiteetin sovelluksessa ja suorittaa pahantahtoisia toimia. (OWASP Top 10 2013,5.)

5.3 Cross-site Scripting (XSS)

Cross-site scripting –haavoittuvuus eli XSS on yksi vakavimmista ja yleisimmistä haavoittuvuuksista web-sovelluksissa. XSS:ssä on kyse käyttäjän web-sovellukseen syöttämän datan liian vähäisestä tai kokonaan puuttuvasta käsittelystä. Puutteellisesti käsitelty data voi pahimmillaan aiheuttaa muutoksia sovelluksessa tai sen ulkoasussa.

Web-sovelluksen lähdekoodi on kuin mitä tahansa koodia ja se rakentuu kirjaimista ja erikoismerkeistä. Koodin rakenne on hyvin herkkää ja yksikin ylimääräinen merkki voi aiheuttaa virheitä. Kun web-sovellukseen lisätään käyttäjän mahdollisuus syöttää dataa, se synnyttää ongelmia. Käyttäjä ei aina käytä sovellusta juuri siten kuten kehittäjät tarkoittivat ja voivat vahingossa tai tahallaan syöttää tekstikenttiin sellaista sisältöä, joka suodattamattomana aiheuttaa ongelmia.

Mikäli käyttäjän syöttämiä erikoismerkkejä ei käsitellä sovelluksessa millään tavoin se johtaa ongelmiin. Tämä johtuu siitä, että sovellus käsittelee käyttäjän syöttämän tekstin virheellisesti osana sovellusta. Muodostamalla syötettä, joka rikkoo sovelluksen koodin oikeasta kohdasta, hyökkääjä voi pystyä ajamaan haavoittuvassa sovelluksessa omaa koodiaan, jolla voidaan kaapata uhrin istunto, muuttaa verkkosivun ulkomuotoa, ohjata käyttäjiä pahantahtoisille sivuille tai muuta haitallista. Alla olevassa kuvassa on havainnollista esimerkki siitä, miten cross-site scripting haavoittuvuus toimii yksinkertaisella tasolla.

Oletetusti web-sovelluksessa on toiminto, jolla käyttäjä voi kirjoittaa viestin näkyville muille käyttäjille. Kuvassa uusi käyttäjä on kirjoittanut muille näkyviin viestin ”Moi, olen uusi käyttäjä!”. Tämä on normaalia käyttöä eikä aiheuta harmia. Alempi esimerkki on tapaus jossa pahantahtoinen käyttäjä on kirjoittanut samalla viestitoiminnolla viestin, joka sisältää perinteisen tekstin sijaan koodia, joka ajetaan. Riippuen hyökkääjän taidoista ja tavoitteista, ajettava koodi voi tehdä halutut toiminnot web-sovelluksessa.

```
<html>
<h1>Kommentti</h1>
Moi, olen uusi käyttäjä!
</html>



<html>
<h1>Kommentti</h1>
<script>pahaakoodia();</script>
</html>
```

Kuva 3 Cross-site scripting demonstrointi

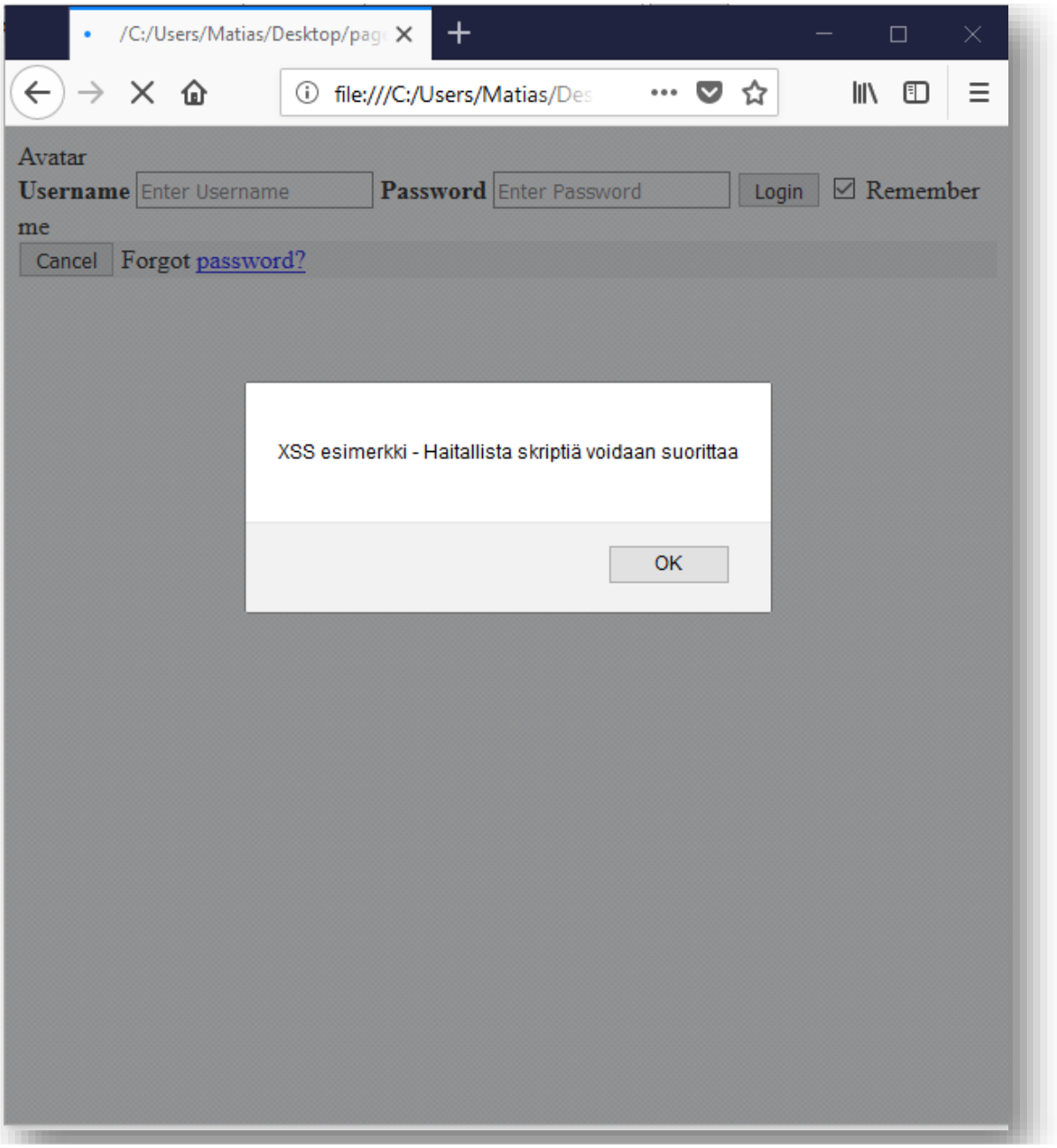
XSS-haavoittuvuuksia on kolmentyyppisiä. Kolme XSS-tyyppiä ovat säilötty (stored), heijastettu (reflected) sekä DOM-pohjainen XSS (document object model). Kaikissa XSS-haavoittuvuuksissa on kuitenkin sama perusidea, jossa lopputuloksena sivustolla ajetaan hyökkääjän luomaa haitallista koodia.

Säilötty XSS-haavoittuvuus aiheutuu siitä, kun haitallinen skripti saadaan pysyvästi osaksi websovellusta. Hyvä esimerkki säilötystä XSS:stä on seuraava. Käyttäjä luo sovelluksen tietokantaan käyttäjätunnuksen, joka sisältää seuraavanlaisen koodikokonaisuuden. (Kuva 4)

```
<script>alert('XSS esimerkki – Haitallista skriptiä voidaan suorittaa')</script>
```

Kuva 4 Esimerkkikoodi

Tästä seuraa se, että haavoittuva sovellus näyttää joka kerta alert-ilmoituksen, kun käyttäjän nimimerkki näkyy sovelluksessa tai latautuu taustalla. Tässä esimerkissä oleva koodinpätkä ei aiheuta varsinaisesti harmia käyttäjälle, mutta havainnollistaa hyvin konkreettista muutosta sivulla, jonka haavoittuvuus saa aikaiseksi. Script –tagien sisään voidaan sijoittaa toiminnallisuuksia jotka keräävät käyttäjän tietoja tai suorittaa toimintoja sovelluksessa. Alla olevassa kuvassa (Kuva 5) on kuvattuna ilmoitus, jonka käyttäjä saa.



Kuva 5 XSS -pop-up

Heijastettu XSS tarkoittaa, että hyökkääjän saa käyttäjän lähettämään sovellukseen dataa, eli tässä tapauksessa haitallista koodia, jonka sovellus ikään kuin heijastaa takaisin käyttäjälle. Hyökkääjän haitallinen data voi olla esimerkiksi linkki, jota painamalla käyttäjä suorittaa hakutoiminnon sivulla. Kuten uuden säilötyn XSS:n käyttäjän rekisteröinnissä, hakutoimintoonkin voi sisällyttää haitallista koodia, jonka tuotos sitten ilmestyy käyttäjälle. Erona säilötyn ja heijastetun XSS:n välillä on kesto, lopputuloksena molemmissa ajetaan haitallista koodia.

DOM-pohjainen XSS on hyökkäys, jossa haitallinen koodi ajetaan ja lopputuloksena DOM-ympäristö muuttuu ja käyttäjän selaimen käsittelemä koodi ajetaan virheellisesti. Sivusto itsessään

ei muutu kuten säilötyssä XSS:ssä, mutta käyttäjänpuolen koodi ajetaan erilaisesti haitallisen koodin tekemistä muutoksista johtuen. (DOM Based XSS 2015.)

XSS-haavoittuvuus voidaan korjata pääosin kolmella tavalla. Sovelluksen tulisi käsitellä käyttäjän siihen syöttämää dataa sillä tavoin, ettei se pääse vaikuttamaan sivuston ulkoasuun. Syötteen käsittely voi olla esimerkiksi enkoodaus. Enkoodaus tarkoittaa, että merkki muutetaan joksikin muuksi, jotta merkkejä ei lueta osaksi sovelluksen koodia. Esimerkkinä enkoodauksesta voidaan käyttää vaikkapa lainausmerkkiä, joka on yleinen merkki web-sivujen teksteissä, mutta jota käytetään myös web-sivun lähdekoodissa. Mikäli verkkosivulla halutaan näyttää lainausmerkki, se tulee enkoodata. Enkoodaus tehdään, jotta lainausmerkki ei vaikuta sivuston rakenteeseen vaan sivusto tietää, että tarkoitus on pelkästään näyttää lainausmerkki. Enkoodaustapoja on erilaisia, web-sovelluksissa käytetään yleensä URL-enkoodausta. URL-enkoodaus muuttaa lainausmerkin merkeiksi %22. Kun merkeille tehdään käänteinen prosessi eli dekoodaus, merkkijono %22 muuttuu takaisin lainausmerkiksi. Tällä tavoin syöte ikään kuin suojataan kuljetuksen ajaksi.

Toinen tapa torjua XSS-hyökkäyksiä on asettaa web-sovelluksen alla toimiva palvelin ohjeistamaan selaimen toimintaa lähettämällä http-vastauksissaan mukana tietynlaisilla otsakkeilla. Tällaisia otsakkeita ovat esimerkiksi Content-Security-Policy ja X-XSS-Protection –otsakkeet. Content-Security-Policy lyhyesti selitettynä määrää hyväksytyt alkuperät, joista selain saa ladata sisältöä sivulle ja X-XSS-Protection –otsake käskee selainta lopettamaan sivun latauksen kun selain huomaa heijastetun XSS-hyökkäyksen. (Web technology for developers 2018b; Web technology for developers 2018c)

5.4 Turvaton viittaus tietoalkioon

Turvaton viittaus tietoalkioon tapahtuu, kun hyökkääjän on esimerkiksi mahdollista muuttaa tiettyyn tietoon tai tiedostoon viittaava arvo http-kutsussa. Oletetaan, että henkilö A kirjoittaa viestiä jalkapallojoukkueita käsittelevällä sivulla sivun keskustelupalstalle. Jalkapallosivulla on rajoitettu keskustelut niin, että vain saman joukkueen jäsenet voivat kirjoittaa oman joukkueensa keskustelupalstalle. Henkilö A kirjoittaa oman joukkueensa palstalle ja huomaa, että hänen lähettämässään http-pyynnössä on sisällytettynä muuttuja joukkue=15. A arvaa arvon 15 edustavan hänen omaa joukkuettaan. Hän kirjoittaa uuden viestin, mutta vaihtaa muuttujan joukkue arvon joksikin muuksi. Hetken kuluttua henkilö B, joka pelaa eri joukkueessa kuin A, voi lukea oman joukkueensa palstalta ulkopuolisen A:n kirjoituksen. Tätä tarkoittaa turvaton viittaus tietoalkioon.

5.5 Turvallisuusasetusten miskonfiguraatiot

Palvelimissa niin kuin myös web-sovelluksissa on mahdollista asettaa monia eri konfiguraatioita, joista jotkin voivat liian salliviksi asetettuina aiheuttaa turvallisuusuhkia. Tärkeimpiä konfiguraatioita, jotka tulisi asettaa oikein, ovat portit ja palomuurisäännöt. Tarpeettomasti

avoinna olevat portit avaavat pääsyn hyökkääjälle ja suurentavat hyökkäyspinta-alaa. Lisäksi, vaikka käytössä olisi palomuuri, joka rajoittaa pääsyä verkkoon palvelimen verkkoon, mutta käytössä ei ole paikallista palomuuria (host-based firewall), on mahdollista päästä käsiksi palvelimeen, mikäli hyökkääjä onnistuu pääsemään samaan verkkoon esimerkiksi huijaamalla tiensä sisään yrityksen tiloihin ja fyysiseen verkkokytkentään.

Väärien konfiguraatioiden välttämiseksi tulee määrittää mitä ominaisuuksia järjestelmä tarvitsee ja niiden mukaisesti määrittää pois päältä ne ominaisuudet, jotka jäävät tarpeen ulkopuolelle. Näin pienennetään web-sovelluksen hyökkäyspinta-alaa. Turhien ominaisuuksien poistamista ja oikeiden konfiguraatioiden asettamista tarpeellisiin komponentteihin kutsutaan koventamiseksi. Säännöllinen järjestelmän koventaminen olisi syytä tehdä toistuvasti säännöllisin väliajoin. (OWASP Top 10 – 2013, 2013.)

Esimerkin toteutukseen tarvittiin ainoastaan selaimen kehittäjätyökaluja. Vierailemalla web-sovelluksessa ja tutkailemalla sen lähettämiä http-vastauksia saadaan selville tietoja palvelimesta. Tässä tapauksessa kyseessä on helposti korjattava miskonfiguraatio, ja lopulliset vaikutukset eivät ole hirvittävän suuret. Silti tämänlaiset tiedot helpottavat hyökkääjien toimintaa. Mikäli hyökkääjä tietää täydelliset versiotiedot, on tämän helppo etsiä versiokohtaisia haavoittuvuuksia internetistä tai keskittää energiansa uusien haavoittuvuuksien etsimiseen.

Näiden tietojen kerääminen on helppo automatisoida, jolloin tietoja paljastava palvelin joutuu todennäköisesti hyökkäyslistalle. Toisaalta jotkin ammattilaiset ovat virallisessa dokumentaatiossa sitä mieltä, että paljastettujen tietojen salaaminen hankaloittaisi erinäisten käyttöön liittyvien ongelmien selvittämistä ja piilotetut http-otsakkeet loisivat vain valheellista turvallisuuden tunnetta. (Reduce or remove server headers 2018.)

Kuitenkin, alla olevassa kuvassa (Kuva 6) on nähtävissä, että palvelin lähettää http-vastauksissaan tiedon olevansa Microsoft IIS 8.5 –palvelin sekä web-sovelluksen oleva ASP.NET pohjainen sovellus.

Kuva 6 Esimerkki miskonfiguraatiosta

5.6 Arkaluonteisen datan paljastuminen

Monet sovellukset nykypäivänä käsittelevät arkaluontoista dataa, kuten asiakastietoja, luottokorttimaksuja ja pankkitoimintaa. Tämän kaltaisen datan paljastuminen aiheuttaa monesti uhreilleen kolhuja maineeseen sekä mahdollisesti taloudellisia tappioita. Arkaluontoisen datan paljastuminen voi tapahtua melkeinpä mitä vain haavoittuvuutta hyväksikäyttämällä. Arkaluontoiseksi dataksi voidaan luokitella myös istuntotunnisteet. Istuntotunnisteita kalastelemalla muita haavoittuvuuksia hyödyntämällä voidaan päästä käsiksi käyttäjän muihin tietoihin.

Arkaluontoinen data pitäisi poikkeuksetta olla säilöttynä salattuna, jolloin vaikka hyökkääjä pääsisi käsiksi itse dataan, se olisi ei-luettavassa muodossa ja näin ollen hyökkääjälle hyödytön ilman tapaa dekryptata se. Istuntotunnisteen turvallisuutta voidaan lisätä asettamalla istuntotunnisteen sisältävälle evästeelle HttpOnly -lippu. HttpOnly -lippu rajoittaa evästeen käyttöä siten, ettei esimerkiksi JavaScript pysty lukemaan sitä.

Arkaluontoisen datan paljastamista ei sinänsä voida tehdä vain yhdellä tavalla vaan kyseessä on enemmänkin seuraus hyökkäyksestä. Mikäli web-sovelluksessa on esimerkiksi XSS-haavoittuvuus, voidaan suojaamattomat evästeet kaapata ja näin päästä käsiksi istuntotunnisteeseen sekä sitä kautta käyttäjän henkilötietoihin.

5.7 Pääsynhallinnan puuttuminen funktiotasolla

Useimmissa web-sovelluksissa vaaditaan sisäänkirjautuminen perinteisen sisäänkirjautumissivun kautta ennen kuin sovellusta pääsee käyttämään. Vaikka sovellus vaatisi sisäänkirjautumista, ei se välttämättä tarkoita että funktiot eli toiminnot olisi rajoitettu vain sisäänkirjautuneille henkilöille. Samanlainen esimerkki on järjestelmänvalvojan toiminnot, joita peruskäyttäjä voi kutsua, kunhan on kirjautunut edes jollakin tunnuksilla järjestelmään.

Korjausehdotus pääsynhallinnan puuttumiseen on yksinkertaisesti pääsynhallinnan implementointi. Sovelluksessa pitäisi olla pääsynhallintamoduli, jota kaikkien toimintojen tulisi käyttää. Hyvä perussääntö on estää pääsy kaikkiin toimintoihin ja sallia toimintoja roolien mukaan. (OWASP Top 10 – 2013 2013.)

Tämä haavoittuvuus voidaan testata peruskäytöllä, mutta on helpompi testattava esimerkiksi Burp Suite – työkalulla. Hyvä keino tähän on käyttää sovellusta ensin sisäänkirjautuneena käyttäjänä ja ohjata liikenne Burp Suiten välityspalvelimeen. Tämän jälkeen tulee uloskirjautua sovelluksesta, mikä lopettaa istunnon sovelluksessa. Burp Suiten historiaosiosta näkee http-kutsut, joita sovellukselle on lähetetty. Näitä kutsuja voi lähettää uudelleen sovellukselle Burp Suiten toiminnallisuuksilla, ja tutkailla mitä sovellus vastaa nyt kun istunto ei ole enää voimassa. Mikäli sovellus suorittaa toimintoja edelleen, sovellus on haavoittuva. Sama voidaan toistaa järjestelmänvalvojan tunnuksilla.

5.8 Cross-Site Request Forgery

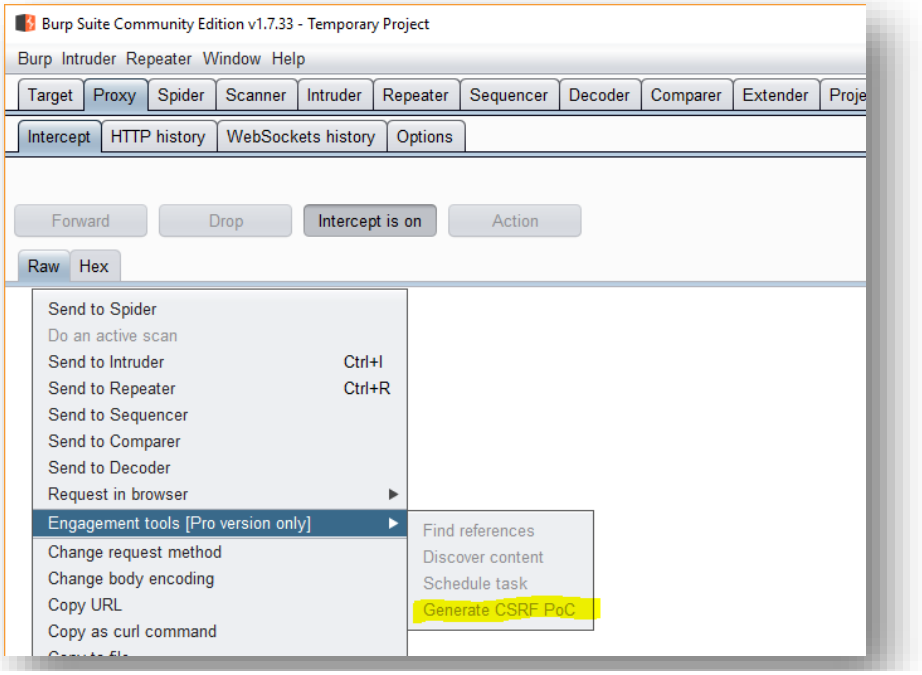
Cross-Site Request Forgery eli CSRF-haavoittuvuus syntyy, kun web-sovelluksen vastaanottamien http-kutsujen alkuperää ei tarkasteta. Tämä johtaa siihen, että kutsut, mukaan lukien haitallisten sivujen lähettämät tarkastamattomat kutsut käsitellään.

Yleisesti http-kutsuja tehdään vain, kun käyttäjä tekee web-sivulla jotakin, esimerkiksi painaa kirjoittaa hakusanan hakupalkkiin ja painaa hakunappia. Samalla sivulla sijaitsevat toiminnallisuudet lähettävät kutsuja joiden alkuperä (engl. origin) on sama kuin sovellus itse. Jotkin haitalliset sivut sisältävät koodia, joka voi lähettää kutsuja haavoittuvaan web-sovellukseen vaikkapa haitallisen sivun nappeja painaessa tai jopa pelkästään haitallisen sivun latautuessa ruudulle. Kun käyttäjä eksyy haitalliselle sivulle ollessaan kirjautuneena sisään haavoittuvaan sovellukseen, haitallinen sivu voi tehdä käyttäjän istuntotunnisteita hyödyntäen kutsuja web-sovellukseen ja suorittaa toimintoja siinä.

Haavoittuva web-sovellus voidaan pelastaa CSRF-haavoittuvuudelta lisäämällä käyttöön niin kutsuttu anti-CSRF token. Anti-CSRF token on muuttuja, joka lisätään jokaiseen http-kutsuun, jonka sovelluksen halutaan huomioivan. Mikäli anti-CSRF tokenia ei ole, sovellus sivuuttaa kutsun.

Anti-CSRF token toimii siten, että palvelin antaa selaimelle muuttujan otsakkeena, jonka selain antaa kutsuihin, jotka sopivat yhteen selaimissa olevan niin kutsutun same-origin-policyn kanssa. Same-origin-policy on selainten peruspilareita, joka määrittelee miten yhdestä alkuperästä ladattu asiakirja tai skripti voi toimia yhdessä toisen alkuperän resurssien kanssa. (Mozilla 2018.)

CSRF-haavoittuvuutta voidaan hyödyntää tekemällä HTML-sivu, joka sisältää toiminnon joka lähettää haavoittuvaan sovellukseen http-kutsun. Kätevin tapa demonstroida tämä haavoittuvuus on Burp Suite ja sen alatyökalu, jolla generoidaan CSRF-todiste-esitys (CSRF proof-of-concept). Burp Suitella on mahdollista generoida http-kutsun perusteella html-sivu, joka sisältää kutsun lähetyksen ja html-koodatun napin jolla kutsun voi lähettää. Käyttäjän kirjauduttua sisään hänen tulee avata generoitu html-sivu, ja painaa nappia. Mikäli sovellus suorittaa toiminnon, jonka alkuperä on haitalliselta sivulta, kyseessä on haavoittuva sovellus.



Kuva 7 Burpin CSRF-työkalu

5.9 Haavoittuvien komponenttien käyttö

Markkinoilla olevat komponentit ja ohjelmistoversiot ovat jatkuvasti hyökkääjien tutkimuksen alaisena. Suosituista ohjelmistoista ja laitteista pyritään löytämään haavoittuvuuksia, joita

kehittäjät eivät ole ehtineet tai huomanneet korjata ennen julkaisua. Haavoittuvuuksien löytyminen tarkoittaa hyökkääjille rahaa tai muutoin hyvää mieltä motiiveista riippuen. Kehittäjille haavoittuvuuden löytyminen tarkoittaa kuitenkin töitä ja haavoittuvuuden paikkaavan ohjelmistopäivityksen kehittämistä eli alan kielellä patchausta (engl. patching).

On usein vain ajan kysymys, että haavoittuvuuksia löytyy. Tästä syystä on tärkeää päivittää ohjelmistot viimeisimpiin versioihin ja korvata laitteet, joista löytyy haavoittuvuuksia, joita ei pystytä remedioimaan. Monissa järjestelmissä on mahdollisuus asettaa automaattiset päivitykset voimaan, jolloin uusimmat päivitykset haetaan automaattisesti ilman käyttäjän erillistä hyväksyntää jokaiselle päivitykselle.

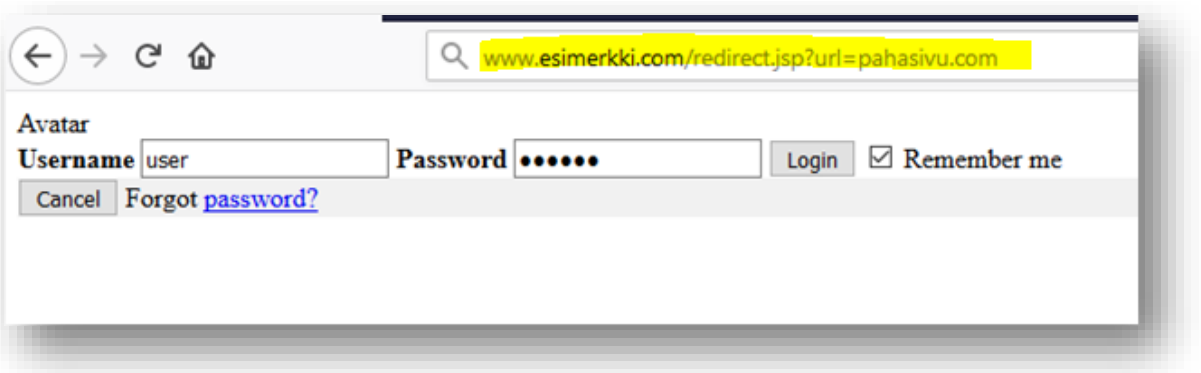
Haavoittuvat komponentit tunnistaa usein vain pysymällä ajan tasalla löytyneistä haavoittuvuuksista. On syytä pitää päivitettävissä olevat komponentit päivitettyinä sekä huolehtia että käytössä olevien komponenttien elinikää on vielä jäljellä. Usein taloudellisista syistä tuotteiden kehittäjät hylkäävät jossakin vaiheessa vanhentuneet tuotteet eivätkä enää kehitä uusia päivityksiä niihin. Tämä johtaa siihen, että vanhoja vikoja ei enää korjata ja haavoittuvuudet jäävät olemaan ja tuotteen omistajat ovat uhreja löytyneille haavoittuvuuksille.

5.10 Vahvistamattomat uudelleenohjaukset ja edelleen lähetys

Vahvistamattomissa uudelleenohjauksissa ja edelleen lähetyksissä on kyse siitä, että kohdesivulle vievään linkkiin lisätään parametri url, joka ohjaa linkkiä klikkaavan uhrin haitalliselle sivulle. Haasteena tämän haavoittuvuuden hyväksikäyttöön on se, että saadaan uhri klikkaamaan haitallista linkkiä. Linkki voidaan sisällyttää vaikkapa roskapostiin, jonka pahaa-aavistamaton uhri sitten avaa ja klikkaa linkkiä.

OWASPin mukaan ensimmäinen tapa on välttää uudelleenohjauksien käyttöä kokonaan. Toinen tapa on jättää pois käyttäjän syöttämät parametrit. (OWASP Top 10 - 2013, 16.)

Tätä haavoittuvuutta voidaan hyväksikäyttää lisäämällä sovelluksen URL:iin loppuun uudelleenohjaus ja tutkia ohjaako sovellus käyttäjän eteenpäin haitalliselle sivulle vai ei. Uudelleenohjaus voidaan lisätä esimerkiksi tavalla, joka on esitettynä kuvan (Kuva 8) osoitepalkissa. Mikäli sovellus ohjaa pahasivu.comiin, kyseessä on haavoittuva sovellus.



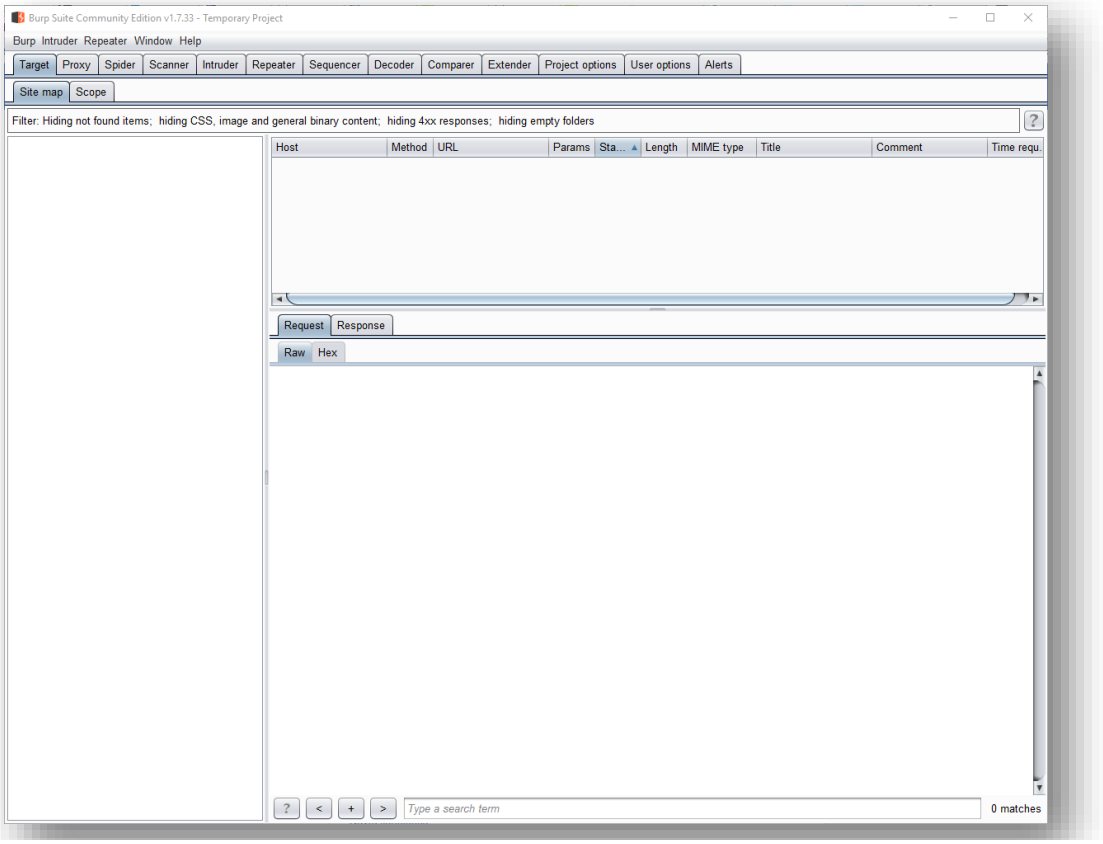
Kuva 8 Uudelleenohjaus

6 Työkalut

Tässä kappaleessa käsitellään web-sovellusten testaukseen soveltuvia työkaluja sekä niiden ominaisuuksia. Käsiteltävät työkalut ovat yleisesti käytössä alan harrastajilla ja ammattilaisilla. Niiden käyttö on suositeltavaa, sillä niiden kautta on mahdollista automatisoida tehtäviä, jotka manuaalisesti kirjoitettuina veisivät paljon aikaa. Lisäksi ne tarjoavat mittavan määrän tietoa testattavasta ympäristöstä, kun niitä käytetään oikein.

6.1 Burp Suite

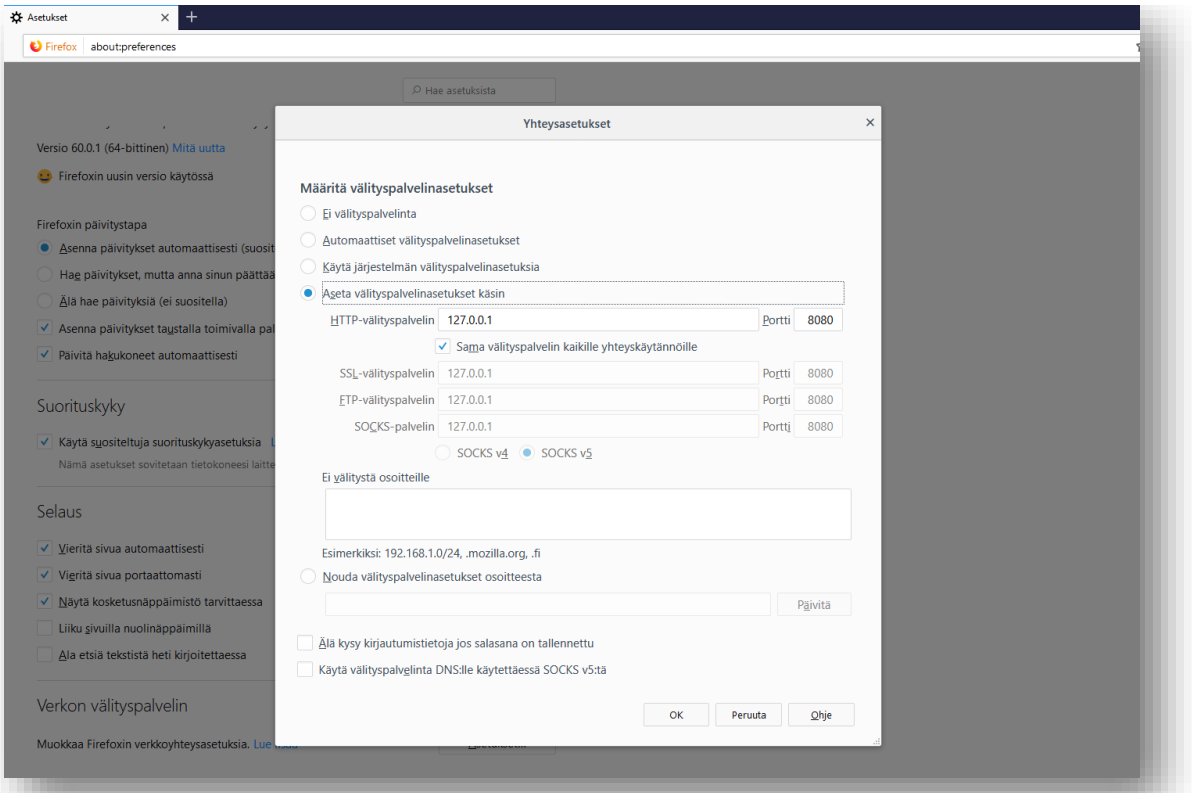
Burp Suite on web-sovellusten tietoturvatestaukseen luotu monipuolinen työkalu, josta on saatavilla ilmainen Community Edition ja lisäksi maksullinen ammattikäyttöön tarkoitettu Professional Edition.



Kuva 9 Burpin käyttöliittymä

Keskeisimpiä ominaisuuksia Burp Suitessa on sen välityspalvelimena toimiminen, jolloin testaaja voi lähettää http-kutsun web-sovelluksesta, keskeyttää sen kulun välityspalvelimella ja muokata sen sisältöä. HTTP -kutsun muokkaaminen mahdollistaa erinäisten haavoittuvuuksien löytämisen sekä niiden hyödyntämisen.

Burpin välityspalvelin toimii kuten muutkin välityspalvelimet ja sen käyttö on helppo aloittaa esimerkiksi Mozilla Firefoxilla. Burp pystyy generoimaan oman sertifikaatin, jonka voit viedä Firefoxin sallittujen sertifikaattiauktoriteettien listaan, jolloin selain ei turhaan luule Burpin kautta kulkevaa liikennettä pahantahtoiseksi. Tämän lisäksi sinun tulee säätää Firefoxin välityspalvelinasetukset käyttämään Burpin välityspalvelinta kaikissa protokollissa. Kuvassa (Kuva 6) on määritelty Firefoxin välityspalvelinasetukset toimimaan Burpin kanssa.



Kuva 10 Firefoxin välityspalvelinasetukset

Burp sisältää monia eri ominaisuuksia, joista suurin osa on rajoitettuna pois ilmaisversiossa. Tämän lisäksi käyttäjäkunnan on mahdollista luoda omia lisäosia Burpiin ja jakaa ne muiden Burpin käyttäjien saataville. Burpissa on tällä hetkellä useita kymmeniä ilmaisia, yhteisön luomia lisäosia. (BApp Store 2018.)

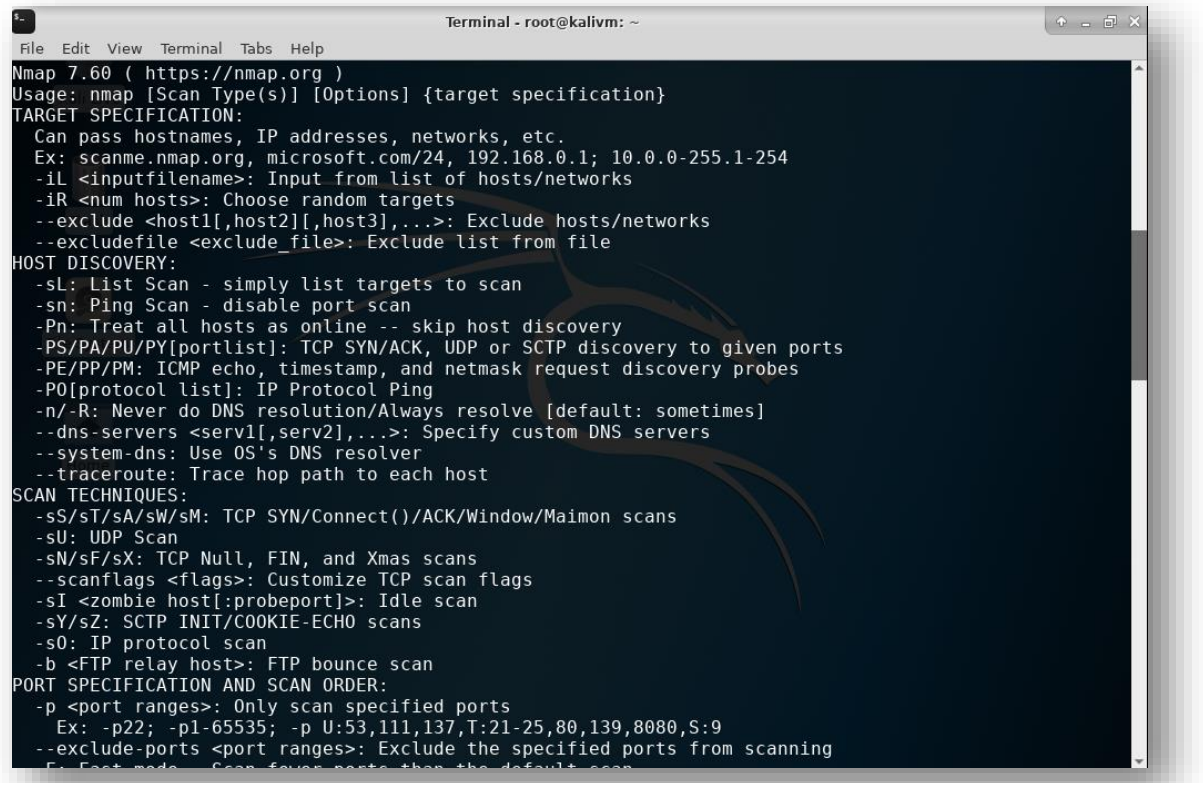
6.2 Nessus

Nessus on Tenablen tarjoama työkalu verkkotason haavoittuvuusskannausten tekemiseen ja sitä voidaan käyttää myös web-sovellusten skannaamiseen. Nessus on käytössä miljoonilla käyttäjillä ympäri maailmaa ja sen avulla voidaan tunnistaa haavoittuvuuksia, miskonfiguraatioita sekä haittaohjelmia. Nessuksen skannauksia voidaan helposti räätälöidä kohdejärjestelmästä riippuen sen monipuolisten valikkojen ansiosta. Nessus on maailman suosituin haavoittuvuusskanneri ja siksi tutustumisen arvoinen työkalu. (Nessus Professional 2018.)

6.3 Nmap

Nmap on komentoriviltä eli terminaalista käytettävä työkalu, jolla voidaan kartoittaa kohdejärjestelmästä esimerkiksi avoimia portteja sekä kohteessa käytössä oleva/ käyttöjärjes-

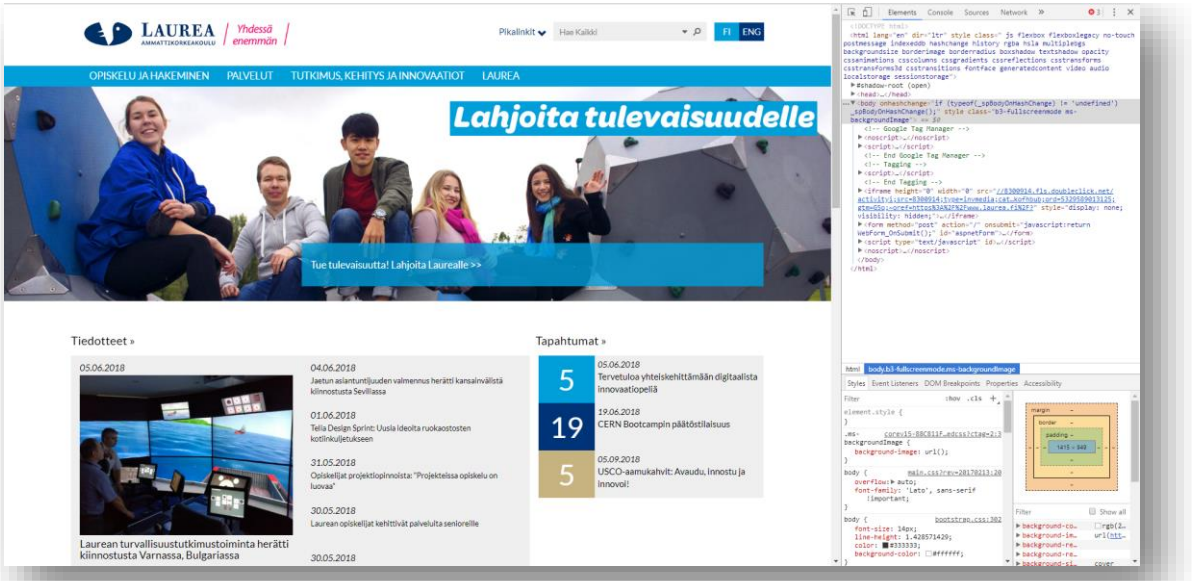
telmä. Nmap on vakiona sisällytetty Kali Linux -käyttöjärjestelmään, joka on Debian -pohjainen, erityisesti penetraatiotestaukseen kehitetty Linux-distribuutio. Kali Linux on vapaasti ladattavissa ja sisältää monia hyödyllisiä työkaluja Nmapin lisäksi.



Kuva 11 Nmapin näkymä terminaalissa

6.4 Selainten kehittäjätyökalut

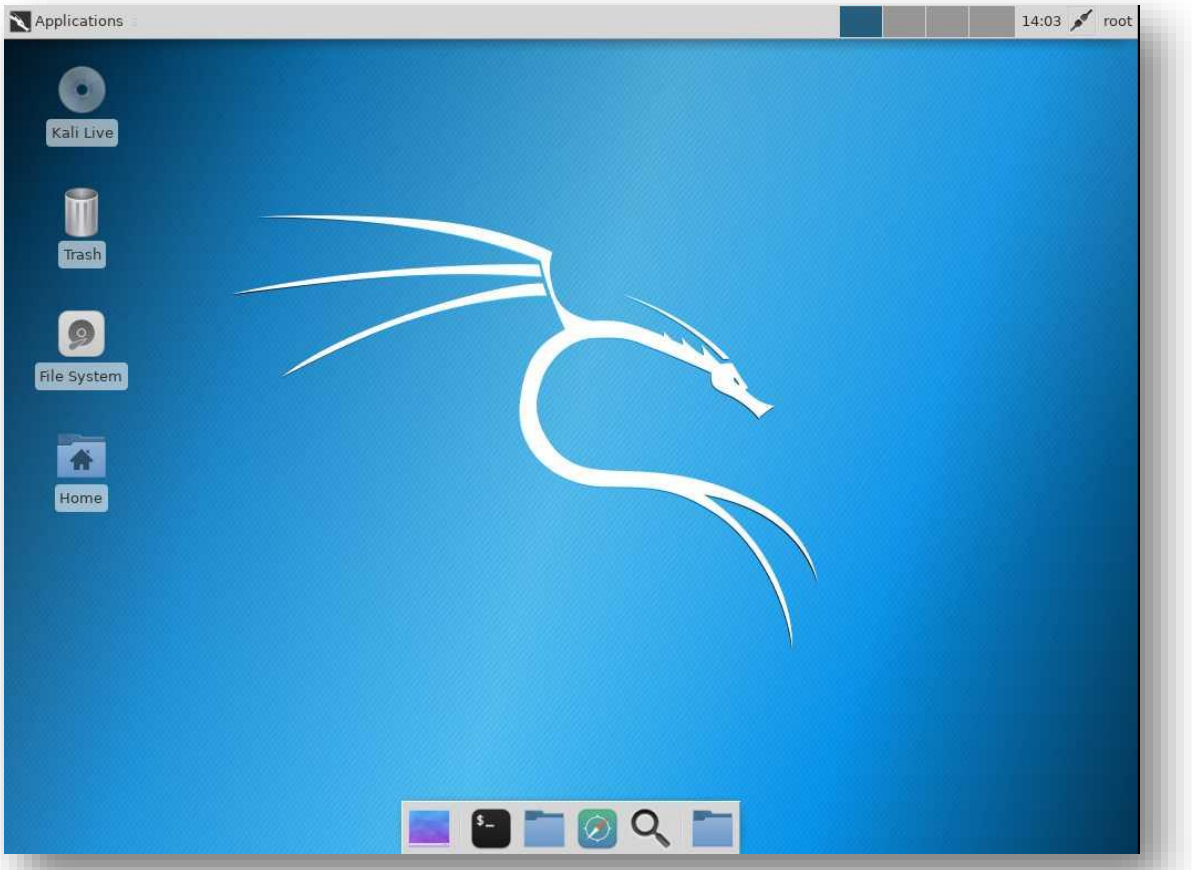
Yleisimmissä selaimissa, kuten Google Chromessa, Mozilla Firefoxissa ja Internet Explorerissa on saatavilla F12-näppäintä painamalla avautuvat kehittäjätyökalut, joilla pystyy tutkimaan verkkosivujen rakennetta sekä toiminnallisuuksia tarkemmin kuin pelkällä verkkosivujen selaamisella. Näitä kehittäjätyökaluja käyttämällä on mahdollista saavuttaa tarkempi kuva web-sovelluksen toimintaperiaatteista sekä huomata mahdollisia heikkouksia web-sovelluksen toteutuksessa. Selaimissa on mahdollista myös käyttää toimintoa, jolla voidaan tarkastella verkkosivun lähdekoodia, joka pahimmissa tapauksissa paljastaa kriittisiä tietoja, joita sovelluksen kehittäjät ovat jättäneet kommenteiksi koodiin.



Kuva 12 Google Chromen kehittäjätyökalut

6.5 Kali Linux

Kali Linux on nimensä mukaisesti Linux-distribuutio, joka on pitkälti omistettu tietoturvatestauksen suorittamiseen ja eettiseen hakkerointiin. Sen kehitys perustuu avoimeen lähdekoodiin. Kali sisältää monta penetraatiotestaukseen tarkoitettua työkalua esiasennettuna ja on tästä syystä suosittu työkalu niin tietoturva-ammattilaisten kuin harrastelijoidenkin keskuudessa. Nimenomaan web-sovelluksille tarkoitettuja työkaluja Kali Linux sisältää yli 40 kappaletta. Tässä opinnäytetyössä käsitellyt työkalut löytyvät kaikki esiasennettuina Kali Linuxin uusimmasta versiosta. (Kali Linux 2018.)



Kuva 13 Kali Linux aloitusnäkymä

7 Yhteenveto

Tämän opinnäytetyön päätarkoitus oli tutkia web-sovelluksiin kohdistuvat yleisimmät, vaarallisimmat ja ajankohtaisimmat haavoittuvuudet. Tarkoitus oli myös käydä läpi ja esitellä web-sovellusten testauksessa käytettyjä työkaluja. Tämän lisäksi tavoitteena oli luoda materiaaleja, joita voidaan käyttää itseopiskelussa. Materiaalit sopivat erityisesti web-sovellusten tietoturvatarkastamisen opiskeluun.

Keskeisin tuotos on sisällytetty tähän opinnäytetyöhön, joka sisältää kymmenen suurimman haavoittuvuuden yleisselvityksen, korjausehdotukset ja esimerkkitapauksen. Tuotos on hyödynnettävissä web-sovelluksen tietoturvatestaajan perustaitojen kartuttamisessa. Tuotoksen arvo on merkittävä pienyrityksille, joiden ei ole mahdollista käyttää varoja tietoturvaan.

Käytettyjä menetelmiä voidaan kehittää jatkossa alan jatkuvien muutosten vuoksi ja se on myös hyvin suositeltavaa, jotta menetelmät pysyvät ajan tasalla. Jatkokehitys on mahdollista

myös OWASPin materiaalin ja alan ajankohtaisimpien haavoittuvuuksien uusiutuessa muutaman vuoden välein. Sovellusten kehitystyön edetessä syntyy myös uusia haavoittuvuuksia, mikä aiheuttaa tietoturva-alalla jatkuvia muutoksia ja tekee alasta ajankohtaisen aina.

Lähteet

Painetut

Studdard, D. & Pinto, M. 2011. The Web Application Hacker's Handbook 2nd edition. Indianapolis: John Wiley & Sons, Inc.

Sähköiset

OWASP Top 10 – 2013. 2013. OWASP Foundation. Viitattu 5.6.2018. https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/owasptop10/OWASP%20Top%2010%20-%202013.pdf

Cross site scripting –haavoittuvuudet. 2008. Viestintävirasto. Viitattu 5.6.2018. https://www.viestintavirasto.fi/kyberturvallisuus/tietoturvanyt/2008/03/P_6.html

Web technology for developers. 2018. Mozilla and individual contributors. Viitattu 1.6.2018. https://developer.mozilla.org/en-US/docs/Web/

Kali Linux. 2018. Offensive Security. Viitattu 2.6.2018. https://www.kali.org/about-us/

Nessus Professional. 2018. Tenable. Viitattu 5.6.2018. https://www.tenable.com/products/nessus/nessus-professional

A timeline of the Ashley Madison hack. 2017. Digital Guardian. Viitattu 5.6.2018. https://digitalguardian.com/blog/timeline-ashley-madison-hack

Reduce or remove server headers. 2018. Tunetheweb. Viitattu 5.6.2018. https://www.tunetheweb.com/security/http-security-headers/server-header/

Introduction to SQL. 2018. W3Schools. Viitattu 5.6.2018. https://www.w3schools.com/sql/sql_intro.asp

DOM Based XSS. 2018. OWASP Foundation. Viitattu 6.6.2018. https://www.owasp.org/index.php/DOM_Based_XSS

SQL query structure. 2013. IBM Knowledge Center. Viitattu 11.6.2018. https://www.ibm.com/support/knowledgecenter/en/SSGU8G_12.1.0/com.ibm.dbdk.doc/ids_dbdk_028.htm

BApp Store. 2018. PortSwigger. Viitattu. 18.6.2018. https://portswigger.net/bappstore

Kuvat

Liitteet

Liite 1: Termien selitykset

Web – Webillä tarkoitetaan verkkoa eli internetiä. Internet on laaja tietokoneiden ja muiden laitteiden verkko.

CIA-triadi – CIA triadi on periaate, jonka mukaan tietoturva jaetaan kolmeen pääosaan: luottamuksellisuus, eheys ja saatavuus (engl. Confidentiality, Integrity, Availability).

Remediointi – vahingon peruuttaminen tai pysäyttäminen, tarkoittaa tässä yhteydessä myös ratkaisun laatimista jotta tulevaisuudessa voitaisiin välttyä kyseiseltä vahingolta.

http-viesti – http-kutsut ovat tapa välittää dataa palvelimen ja clientin välillä. On olemassa http-kutsuja ja http-vastauksia. Kutsut ovat http-viestejä, jotka lähetetään palvelimelle ja vastaukset ovat palvelimen takaisin lähettämiä http-viestejä. (Web technology for developers 2018.)

Haavoittuvuus – Asia, joka mahdollistaa väärinkäytön. Web-sovellusten yhteydessä tarkoittaa usein jotakin teknistä yksityiskohtaa, jonka avulla sovellusta voidaan väärinkäyttää.

Web-sovellus – Verkon kautta saatavilla oleva palvelu. Sijaitsee fyysisesti palvelimella jossakin päin maailmaa, johon käyttäjä ottaa internetin kautta yhteyden.

Tietoturva – Turvallisuuden ala, joka keskittyy tiedon luottamuksellisuuden, eheyden sekä saatavuuden säilyttämiseen.

Palvelunestohyökkäys – Palvelun saattaminen sellaiseen tilaan, jossa sitä ei voida käyttää ja tiedon saatavuus ei ole turvattu. Palvelun estyminen voidaan saavuttaa eri keinoin.

Enkoodaus – Tarkoittaa merkkijonon muuntamista koodattuun muotoon.

Kryptaus ja dekryptaus – Tiedon muuttamista ei-luettavaan muotoon, jolloin tieto pysyy salattuna. Voidaan tehdä eri metodeilla. Kryptaus tarkoittaa luettavan tiedon salausta ja dekryptaus salatun tiedon luettavaan muotoon saattamista.

Internet-of-things, IoT – Esineiden ja asioiden kuten päätelaitteiden, ajoneuvojen, kodinkoneiden sekä muiden sulautettujen laitteden muodostama verkko, joka mahdollistaa laitteiden välisen datansiirron.